**Common Details**

There is one master html file that is leveraging the jQuery Mobile technique of bouncing between divs with a page ID. I refer to each page ID as a screen in the below document.

**User Login Screen (Mobile)**

**Description**

Email – text entry

Password – text entry (password type)

Login – form input button

Register – button

Forgot Password? – button

**On page load:**

Set Local Storage question ID to 0

Set local storage status to new

Check local storage for email and hashed password

If they do exist: put the values in their respective fields and submit the form with auto set to ‘y’

If they don’t exist: open up the page and set auto to ‘n’

**On Login button:**

Check that the email is valid and the password field is not blank

If Email not valid: Raise notification – “Email not valid”

If Password blank: Raise notification – “Password required to login”

Make an AJAX call and POST data:

email=(email text)&password=(password text)&auto=(‘y’ or ‘n’)

**Server side:**

Retrieve the variables from the posted data

If auto is set to ‘n’: Hash the password

Check if the email exists in the database

If it doesn’t exist: Return JSON – {error: ”No Account”}

If the database call fails: Return JSON – {error: Database Error}

Check if the hashed password matches the currently held hashed password for that email

If it doesn’t exist: Return JSON – {error: ”Password Failed”}

If the database call fails: Return JSON – {error: Database Error}

Return JSON – {email: email text, password: hashed password}

**Client side on return:**

If error:

If error == “No Account”:

Raise notification – “User does not exist, please Register”

If error == “Password Failed”:

Raise Notification – “Password not found, please use the forgot password service”

Else:

Raise Notification – Database Error

Else:

set local storage to the email and hashed password that are returned

Move to the Quiz Login Screen

**On Register button:**

Move to the Registration Screen

**On Forgot Password? Button:**

Move to the Forgot Password Screen

**Forgot Password Screen (Mobile)**

**Description**

Email – text entry

Change Password – form input button

**On Change Password button:**

Check email for validity

If Email not valid: Raise notification – “Email not valid”

Make an AJAX call and POST data:

email=(email text)

**Server side:**

Retrieve the variables from the posted data

Check if the email exists in the database

If it doesn’t exist: Return JSON – {error: ”No Account”}

If the database call fails: Return JSON – {error: Database Error}

Else:

Update the password to a common pass, e.g. “temp\_pass”

If the database call fails: Return JSON – {error: Database Error}

Send email to the email address

If the email fails: Return JSON – {error: Email Error}

Email contains link with email POST variable

Return JSON – {success:”y”}

**Client side on return:**

If error:

If error == “No Account”:

Raise notification – “User does not exist, please Register”

Else:

Raise Notification – Database or Email Error

If success == “y”:

Raise notification – “Details have been sent to your email”

Return to the User Login Screen

**Change Password (Web)**

**Description**

Email – text (from POST variable)

Password – text entry (password type)

Repeat Password – text entry (password type)

Change Password – form input button

**On Change Password Button:**

Check email for validity, password if blank, and repeat password if matches password

If Email not valid: Raise notification – “Email not valid”

If Password blank: Raise notification – “Password required to login”

If Repeat Password not equal to Password: Raise notification – “Passwords do not match”

Make an AJAX call and POST data:

email=(email text)&password=(password text)

**Server side:**

Retrieve the variables from the posted data

Check if the email exists in the database

If it doesn’t exist: Return JSON – {error: ”No Account”}

If the database call fails: Return JSON – {error: Database Error}

Else:

Check that the password is temp\_pass

If it isn’t: Return JSON – {error: ”Not Authorized”}

If the database call fails: Return JSON – {error: Database Error}

Else:

Update the password to POSTED password

If the database call fails: Return JSON – {error: Database Error}

Return JSON – {success:”y”}

**Client side on return:**

If error:

If error == “No Account”:

Raise notification – “User does not exist, please Register”

If error == “Not Authorized”:

Raise notification – “Process Failed, request new password from Mobile”

Else:

Raise Notification – Database Error

If success == “y”:

Raise notification – “Password Updated, Please log in”

**Registration Screen (Mobile)**

**Description**

Email – text entry

Password – text entry (password type)

Repeat Password – text entry (password type)

Male – radio

Female – radio

Birthday – date selector

Where do you live? – drop down list selector of counties

Receive Email Notifications? – check box

Register – form input button

**On Register button:**

Check email for validity, password if blank, and repeat password if matches password

If Email not valid: Raise notification – “Email not valid”

If Password blank: Raise notification – “Password required to login”

If Repeat Password not equal to Password: Raise notification – “Passwords do not match”

Make an AJAX call and POST data:

email=(email text)&password=(password text) &male=(1 or 0)&female=(1 or 0)&birthday=(date or all 0’s)&county=(one of list)&update=(1 or 0)

**Server side:**

Retrieve the variables from the posted data

Hash password

Check if the email exists in the database

If it does exist: Return JSON – {error: ”Already Registered”}

If the database call fails: Return JSON – {error: Database Error}

Else:

Update the user table with all information that came through

If the database call fails: Return JSON – {error: Database Error}

Return JSON – {email: email text, password: hashed password}

**Client side on return:**

If error:

If error == “Already Registered”:

Raise notification – “User already exists, please login again or use the forgotten password button”

Else:

Raise Notification – Database Error

Else:

set local storage to the email and hashed password that are returned

Move to the Quiz Login Screen

**Quiz Login Screen (Mobile)**

**Description**

Quiz Code – text entry

Join Quiz – form input button

**On Change Password button:**

Check quizcode if blank

If blank: Raise notification – “Quiz Code needed to proceed”

Make an AJAX call and POST data:

quizcode=(quizcode text)&password=(hashed password)

**Server side:**

Retrieve the variable from the posted data

Check if the quizcode exists in the database

If it doesn’t exist: Return JSON – {error: ”No Quiz”}

If the database call fails: Return JSON – {error: Database Error}

Check if user is already has a participant table

If not: create a participant table for the user

If the database call fails: Return JSON – {error: Database Error}

Check if user is a participant for that quiz

If not: add a line to participant table joining the quiz

If the database call fails: Return JSON – {error: Database Error}

Return JSON – {success:”y”}

**Client side on return:**

If error:

If error == “No Quiz”:

Raise notification – “Quiz has not been registered talk to the quiz master”

Else:

Raise Notification – Database Error

If success == “y”:

Move to the Quiz Wait screen

**Quiz Wait Screen (Mobile)**

**Description**

Rotating Divs:

Advertisement – image

Correct Answer was… – html text

Current Score – html text

Leader Board – table text

Sign Out – button

**On page load:**

Start a process that checks the database every second

(to be superseded by process that accepts push messages)

Check local storage for status variable

If status new:

Advertising

If status released:

Rotate between:

Last answer correct answer

Current score

Advertising

Leader board

If status intermission:

Rotate between:

Advertising

Leader board

If status finished:

Rotate between:

Advertising

Leader board

Also add an exit quiz button to divs: clicking will exit quizzically

**On set interval:**

Pause interval

Make an AJAX call and POST data:

quizcode=(quizcode text)&password=(hashed password)

**Server side:**

Retrieve the variables from the posted data

Select question details

If the database call fails: Return JSON – {error: Database Error}

Return JSON – {category:(category), score:(score), question:(question), ansA:(AnsA), ansB:(AnsB), ansC:(AnsC), ansD:(AnsD), questionID:(questionID), status:(quiz\_status)}

Note: Answers need to be randomly set for each user so ansA – ansD consist of CORRECT\_ANSWER, WRONG\_ANSWER\_1, WRONG\_ANSWER\_1, and WRONG\_ANSWER\_1

**Client side on return:**

If error:

Raise Notification – Database Error

If questionID is equal to question id in local storage:

Store status in local storage

Set interval

Else:

Store Question ID in Local Storage

Store status in local storage

Move to Quiz Question Screen

**Quiz Question Screen (Mobile)**

**Description**

Category – html text

Score – html text

Timer Bar – slider bar controlled by javascript (30 seconds)

Question – html text

Answer A – button

Answer B – button

Answer C – button

Answer D – button

**On Answer button:**

Make an AJAX call and POST data:

password=(hashed password)&answer=(A, B, C, or D)&timer=(timer value)&questionID=(questionID)

**Server side:**

Retrieve the variables from the posted data

Check answer == correct answer for questionID:

If correct add timer value to score

If the database call fails: Return JSON – {error: Database Error}

Return JSON – {score:(current score), correct:(last correct score), first:(first score), second:(second score), third:(third score), fourth:(fourth score), fifth:(fifth score), place:(which place you are in)}

Note: current scores are coming from C\_INTERACTION table.

**Client side on return:**

If error:

Raise Notification – Database Error

Else:

Move to Quiz Wait Screen

**User Login Screen (Web)**

**Description**

Email – text entry

Password – text entry (password type)

Login – form input button

Register – button

Forgot Password? – button

**On page load:**

Check local storage for email and hashed password

If they do exist: put the values in their respective fields and submit the form with auto set to ‘y’

If they don’t exist: open up the page and set auto to ‘n’

**On Login button:**

Check that the email is valid and the password field is not blank

If Email not valid: Raise notification – “Email not valid”

If Password blank: Raise notification – “Password required to login”

Make an AJAX call and POST data:

email=(email text)&password=(password text)&auto=(‘y’ or ‘n’)

**Server side:**

Retrieve the variables from the posted data

If auto is set to ‘n’: Hash the password

Check if the email exists in the database

If it doesn’t exist: Return JSON – {error: ”No Account”}

If the database call fails: Return JSON – {error: Database Error}

Check if the hashed password matches the currently held hashed password for that email

If it doesn’t exist: Return JSON – {error: ”Password Failed”}

If the database call fails: Return JSON – {error: Database Error}

Return JSON – {email: email text, password: hashed password}

**Client side on return:**

If error:

If error == “No Account”:

Raise notification – “User does not exist, please Register”

If error == “Password Failed”:

Raise Notification – “Password not found, please use the forgot password service”

Else:

Raise Notification – Database Error

Else:

set local storage to the email and hashed password that are returned

Move to the Quiz Home Screen

**On Register button:**

Move to the Registration Screen

**On Forgot Password? Button:**

Move to the Forgot Password Screen

**Forgot Password Screen (Web)**

**Description**

Email – text entry

Change Password – form input button

**On Change Password button:**

Check email for validity

If Email not valid: Raise notification – “Email not valid”

Make an AJAX call and POST data:

email=(email text)

**Server side:**

Retrieve the variables from the posted data

Check if the email exists in the database

If it doesn’t exist: Return JSON – {error: ”No Account”}

If the database call fails: Return JSON – {error: Database Error}

Else:

Update the password to a common pass, e.g. “temp\_pass”

If the database call fails: Return JSON – {error: Database Error}

Send email to the email address

If the email fails: Return JSON – {error: Email Error}

Email contains link with email POST variable

Return JSON – {success:”y”}

**Client side on return:**

If error:

If error == “No Account”:

Raise notification – “User does not exist, please Register”

Else:

Raise Notification – Database or Email Error

If success == “y”:

Raise notification – “Details have been sent to your email”

Return to the User Login Screen

**Registration Screen (Web)**

**Description**

Email – text entry

Password – text entry (password type)

Repeat Password – text entry (password type)

Organization Name – text entry

Where are you based? – drop down list selector of counties

Receive Email Notifications? – check box

Register – form input button

**On Register button:**

Check email for validity, password if blank, and repeat password if matches password

If Email not valid: Raise notification – “Email not valid”

If Password blank: Raise notification – “Password required to login”

If Repeat Password not equal to Password: Raise notification – “Passwords do not match”

Make an AJAX call and POST data:

email=(email text)&password=(password text)&repeat=(repeat password text)&name=(name of organization)&county=(one of list)&update=(1 or 0)

**Server side:**

Retrieve the variables from the posted data

Hash password

Check if the email exists in the database

If it does exist: Return JSON – {error: ”Already Registered”}

If the database call fails: Return JSON – {error: Database Error}

Else:

Update the user table with all information that came through

If the database call fails: Return JSON – {error: Database Error}

Return JSON – {email: email text, password: hashed password}

**Client side on return:**

If error:

If error == “Already Registered”:

Raise notification – “User already exists, please login again or use the forgotten password button”

Else:

Raise Notification – Database Error

Else:

set local storage to the email and hashed password that are returned

Move to the Quiz Home Screen

**Quiz Home Screen (Web)**

**Description**

Profile – menu link

Quiz – menu link

Purchase – menu link

**On Profile link:**

Move to Profile Screen

**On Quiz link:**

Move to Quiz Build Screen

**On Purchase link:**

Move to Purchase Screen

**Profile Screen (Web)**

**Description**

Email – text entry (with email value inserted)

Change Password – button

Organization Name – text entry (with value inserted)

Where are you based? – drop down list selector of counties (With value selected)

Receive Email Notifications? – check box

Back – button

Update – form input button

**On Change Password button:**

Move to Change Password Screen with Posted Email

**On Back button:**

Move to Quiz Home Screen

**On Update button:**

Check email for validity, password if blank, and repeat password if matches password

If Email not valid: Raise notification – “Email not valid”

Make an AJAX call and POST data:

email=(email text)& name=(name of organization)&county=(one of list)&update=(1 or 0)&emailold=(local storage email)

**Server side:**

Retrieve the variables from the posted data

Check if the oldemail exists in the database

If it doesn’t exist: Return JSON – {error: ”User not recognized”}

If the database call fails: Return JSON – {error: Database Error}

Else:

Update the user table with all information that came through

If the database call fails: Return JSON – {error: Database Error}

Return JSON – {email: email text, password: hashed password}

**Client side on return:**

If error:

If error == “User Not Recognized”:

Raise notification – “User doesn’t exist, please login again”

Move to User Login Screen

Else:

Raise Notification – Database Error

Else:

set local storage to the email and hashed password that are returned

Move to the Quiz Home Screen

**Purchase Screen (Web)**

**Description**

Needs to be worked out with payment system and a way of tracking quiz credits

**Quiz Build Screen (Web)**

**Description**

Quizcode – text entry

Quiz Location – text entry

Number of Questions – number entry

Difficulty – drop down selection

Create Quiz – form input button

**On Create Quiz button:**

Check email for validity, password if blank, and repeat password if matches password

If quizcode blank: Raise notification – “quizcode required to create quiz”

If location blank: Raise notification – “location required to create quiz”

If # of Questions blank: Raise notification – “Number of Questions required to create quiz”

If difficulty blank: Raise notification – “Difficulty required to create quiz”

Make an AJAX call and POST data:

email=(email text)&password=(hashed password)&quizcode=(quizcode text)&location=(location text)&number=(number of questions text)&difficulty=(1, 2, 3, 4, or 5)

**Server side:**

Retrieve the variables from the posted data

Confirm if the email and password exist in the database

If they do not exist: Return JSON – {error: ”Login Error”}

If the database call fails: Return JSON – {error: Database Error}

Else:

Check quizcode isn’t currently being used

If it is: Return JSON – {error: “Quizcode Error”}

Update the quiz table with all information that came through

If the database call fails: Return JSON – {error: Database Error}

Set quiz status to s

Return JSON – {status:”s”, difficulty:(difficulty), number:(number of questions), quizcode:(quizcode)}

**Client side on return:**

If error:

If error == “Login Error”:

Raise notification – “Login Error Please Sign in again”

Move to User Login Screen

If error == “Quizcode Error”:

Raise notification – “Quizcode currently being used, pick another”

Else:

Raise Notification – Database Error

Else:

Set local storage to quiz status

Set difficult in local storage

Set number in local storage

Set quizcode in local storage

Move to the Question Screen

**Question Screen (Web)**

**Description (different elements on view for different statuses)**

Leader board (users, scores and answer speed)– html table

Quizcode – html text

Quiz status – html text

Question category – html text

Question – html text

Answer A – html text

Answer B – html text

Answer C – html text

Answer D – html text

Correct Answer – html text

Quiz Start – button

Release question – button

Next question – button

Intermission – button

Home page - button

**On page load:**

Check local storage for status variable

If status start:

Quizcode

Quiz status

Quiz to start message

Quiz start button

If status new:

Quizcode

Quiz status

Leader board

Category

Question

Answer A

Answer B

Answer C

Answer D

Correct Answer

Release Question Button

Intermission Button

If status released:

Quizcode

Quiz status

Leader board

Category

Question

Answer A

Answer B

Answer C

Answer D

Correct Answer

Next Question Button

Intermission Button

If status intermission:

Quizcode

Quiz status

Leader board

Intermission Message

Next Question Button

If status finished:

Quizcode

Quiz status

Leader board

Quiz finished message

Home button

**On Quiz Start button:**

Set local storage current question to 1

Make an AJAX call and POST data:

quizcode=(quizcode text)&password=(hashed password) &status=”n” &difficulty=(difficulty) &current=(current question)

**Server side:**

Retrieve the variables from the posted data

Select a random question using variables

If the database call fails: Return JSON – {error: Database Error}

Return JSON – { category:(category), question:(question), ansA:(AnsA), ansB:(AnsB), ansC:(AnsC), ansD:(AnsD), correct:(correct answer), status:(status), first:(first score), second:(second score), third:(third score), fourth:(fourth score), fifth:(fifth score)}

Note: current scores are coming from C\_INTERACTION table.

**Client side on return:**

If error:

Raise Notification – Database Error

Else:

Set status in local storage

Set JSON Data in html place holders

Restart Quiz Question Screen

**On Release Question button:**

Make an AJAX call and POST data:

quizcode=(quizcode text)&password=(hashed password)&status=”r”

**Server side:**

Retrieve the variables from the posted data

Update the question status

If the database call fails: Return JSON – {error: Database Error}

Return JSON – {status:(status), first:(first score), second:(second score), third:(third score), fourth:(fourth score), fifth:(fifth score)}

Note: The scores are coming from the C\_INTERACTION table

**Client side on return:**

If error:

Raise Notification – Database Error

Else:

Set status in local storage

Restart Quiz Question Screen

**On Next Question button:**

Set local storage current question to current + 1

Make an AJAX call and POST data:

quizcode=(quizcode text)&password=(hashed password) &status=”n” &difficulty=(difficulty) &current=(current question)

**Server side:**

Retrieve the variables from the posted data

Select a random question using variables

If the database call fails: Return JSON – {error: Database Error}

Return JSON – {category:(category), question:(question), ansA:(AnsA), ansB:(AnsB), ansC:(AnsC), ansD:(AnsD), correct:(correct answer), status:(status), first:(first score), second:(second score), third:(third score), fourth:(fourth score), fifth:(fifth score)}

Note: current scores are coming from C\_INTERACTION table.

**Client side on return:**

If error:

Raise Notification – Database Error

Else:

Check current number against number

If ==:

Make an AJAX call and POST data:

quizcode=(quizcode text)&password=(hashed password)&status=”f”

**Server side:**

Retrieve the variables from the posted data

Update the question status and delete the quizcode

If the call fails: Return JSON – {error: Database Error}

Return JSON – {status:(status), first:(first score), second:(second score), third:(third score), fourth:(fourth score), fifth:(fifth score)}

Note: current scores are coming from C\_INTERACTION table.

**Client side on return:**

If error:

Raise Notification – Database Error

Else:

Set status in local storage

Restart Quiz Question Screen

Else:

Set status in local storage

Set JSON Data in html place holders

Restart Quiz Question Screen

**On Intermission button:**

Make an AJAX call and POST data:

quizcode=(quizcode text)&password=(hashed password)&status=”i”

**Server side:**

Retrieve the variables from the posted data

Update the question status

If the database call fails: Return JSON – {error: Database Error}

Return JSON – {status:(status), first:(first score), second:(second score), third:(third score), fourth:(fourth score), fifth:(fifth score)}

Note: current scores are coming from C\_INTERACTION table.

**Client side on return:**

If error:

Raise Notification – Database Error

Else:

Set status in local storage

Restart Quiz Question Screen

**On Home Page button:**

Move to Home Page